Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Žana Hrastovšek

Prototip sistema za obveščanje o izrednih dogodkih z uporabo tehnologije Google Cloud Messaging

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: doc. dr. Damjan Vavpotič

Ljubljana 2014

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V okviru diplomske naloge raziščite, kako z uporabo mobilnih naprav in tehnologije Google Cloud Messaging podpreti obveščanje gasilcev in reševalcev, ki trenutno večinoma uporabljajo pozivnike (pager). Raziščite in predstavite posamezne potrebne tehnologije in načrtujte možno arhitekturo in podatkovni model sistema. Na tej podlagi izdelajte delujoč prototip sistema za obveščanje.

Izjava o avtorstvu diplomskega dela

Spodaj podpisana Žana Hrastovšek, z vpisno številko **63090050**, sem avtor diplomskega dela z naslovom:

Prototip sistema za obveščanje o izrednih dogodkih z uporabo tehnologije Google Cloud Messaging

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelala samostojno pod mentorstvom doc. dr. Damjana Vavpotiča,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 9. septembra 2014 Podpis avtorja:

Zahvaljujem se podjetju LOGOS.SI d.o.o., ki je pripravilo spletno storitev za enostavno pridobivanje podatkov o izrednih dogodkih s spletnega portala SPIN.

Zahvaljujem se vsem, ki so me med pisanjem diplomskega dela podpirali in prenašali.

Zahvaljujem se mentorju.

Kazalo

Slike

SLIKE

Izvorna koda

Seznam uporabljenih kratic

kratica	angleško	slovensko
JVM	Java Virtual Machine	javanski virtualni stroj
API	Application Programming Interface	vmesnik za programiranje aplikacij
Java EE	Java Platform Enterprise Edition	Java platforma, poslovna različica
Java SE	Java Platform Standard Edition	Java platforma, standardna različica
ORM	Object-Relational Mapping	objektno-relacijsko preslikovanje
SQL	Structured Query Language	strukturiran poizvedovalni jezik
SOAP	Simple Object Access Protocol	protokol za izmenjavo XML sporočil
REST	Representational State Transfer	arhitekturni stil izmenjave sporočil
SDK	Software Development Kit	zbirka orodij za razvoj programske opreme
APK	Android application package file	aplikacijski paket za Android
GCM	Google Cloud Messaging	storitev, ki omogoča pošiljanje sporočil iz aplikacijskih strežnikov na Android naprave in prejemanje povratnih sporočil
GPS	Global Positioning System	sistem za globalno pozicioniranje
SPIN	Reporting system for accidents and interventions	sistem poročanja o intervencijah in nesrečah

Povzetek

Obveščevalni sistem gasilcev in reševalcev večinoma temelji na tehnologiji pozivnikov. Ta tehnologija je sicer zanesljiva in preverjena, a že precej zastarela.

Spletna aplikacija Sistema poročanja o intervencijah in nesrečah (SPIN) Uprave RS za zaščito in reševanje ponuja vpogled v izredne dogodke, prijavljene na številki 112.

Odločili smo se izdelati prototip sistema, ki bo s pomočjo podatkov iz sistema SPIN obveščal uporabnike o izrednih dogodkih. Za ta namen smo razvili mobilno aplikacijo, pri tem pa smo uporabili tehnologijo Google Cloud Messaging.

Google Cloud Messaging nam omogoča enostavno in zanesljivo pošiljanje sporočil z našega strežnika na naprave z nameščenim operacijskim sistemom Android in našo aplikacijo. Na ta način lahko pametni telefon združuje še eno napravo, pozivnik.

Pri obveščanju smo upoštevali uporabnikove želje glede vrste, stanja in kraja dogodka, na uporabnikovo željo pa smo posebej upoštevali še en vidik njegovega konteksta, njegovo lokacijo.

Ključne besede: Android, Java EE, Google Cloud Messaging, GPS, Sistem poročanja o intervencijah in nesrečah.

Abstract

The notification system for firemen and rescuers is based on pagers. This technology is reliable and verified but outdated.

The reporting system for accidents and interventions (SPIN) is a web application of the Administration of the Republic of Slovenia for Civil Protection and Disaster Relief. It provides an insight into incidents, reported to the number 911.

We have decided to develop a prototype of a system, that notifies users about the incidents reported to the SPIN system. For that purpose we have used the Google Cloud Messaging technology.

Google Cloud Messaging allows us to send notifications from our server to the Android devices with our application installed and makes it simple and reliable. That way a smart phone can act as a pager.

When notifying we have taken into account user preferences regarding the field, status and location of the incidents and an aspect of user's context, his location.

Keywords: Android, Java EE, Google Cloud Messaging, GPS, Reporting system for accidents and interventions.

Poglavje 1

Uvod

Slovenijo je v začetku leta 2014 doletela naravna katastrofa. Ob hudih vremenskih razmerah so klonila še tako mogočna drevesa, infrastruktura in tudi življenja. Gasilci so v tem času veliko storili za nas, in čas je, da tudi mi naredimo nekaj zanje.

Obveščevalni sistem prostovoljnih gasilskih društev še vedno temelji na zastareli tehnologiji pozivnikov. To je sicer precej zanesljiva rešitev, a vseeno kliče po modernizaciji, če ne izboljšavi. V današnjem svetu je mobilni telefon nepogrešljiv del življenja posameznika, in tudi v prihodnosti bo tako. Telefon nas spremlja vsakodnevno na vsakem koraku, medtem ko dodatno opremo neradi nosimo s seboj. Pametni telefon namreč že združuje napravo za klicanje, poslušanje glasbe, navigacijo, dostop do svetovnega spleta, zabave in še mnogo več. Zakaj ne bi temu dodali še pozivnik in omogočili gasilcem in reševalcem, da so lahko dosegljivi tudi brez dodatne opreme, ostalim uporabnikom pa, da se izognejo morebitnim nevarnim situacijam.

Ministrstvo za obrambo in Uprava RS za zaščito in reševanje sta postavila Sistem za poročanje o intervencijah in nesrečah (SPIN), ki prikazuje aktualne dogodke prijavljene na številki 112 in omogoča naročanje na dogodke preko vira RSS. Sistem je že tako zelo priljubljen med gasilci, želimo pa ga narediti še priročnejšega. Odločili smo se storiti korak naprej in omogočiti obveščanje preko pametnih telefonov z operacijskim sistemom Android s pomočjo tehnologije Google Cloud Messaging.

Za še večjo uporabnost aplikacije pa bomo upoštevali tudi uporabnikov kontekst. Uporabnikov kontekst je sestavljen iz fizičnega, psihičnega, socialnega, in-

formacijskega in tehnološkega. V diplomskem delu se bomo osredotočili na uporabnikov fizični kontekst, ki nosi informacijo o lokaciji uporabnika glede na izredne dogodke.

1.1 Motivacija

Pametni telefoni so zelo pomemben del posameznikovega vsakdana. Dojemamo jih kot osebne asistente, brez katerih smo hitro izgubljeni. Zaradi takšne navezanosti pa je skoraj nemogoče, da spregledamo obvestila, ki jih prejmemo na pametne telefone. Android naprave nas na primer obveščajo o novi elektronski pošti, Facebook obvestilih, Tweetih in posodobitvah samih aplikacij. Vse to pa omogoča tehnologija Google Cloud Messaging, s pomočjo katere lahko iz drugih naprav in aplikacij dosežemo mobilne naprave z operacijskim sistemom Android. Uporabnik oziroma naročnik je tako obveščen o dogodku brez potrebe po poizvedovanju. Tehnologija poskrbi za zanesljivo in učinkovito posredovanje obvestil.

1.2 Opis problema

Zaradi hitrega tempa življenja in informacij, ki so tako rekoč na pladnju, se počutimo obremenjeni, če jih še vedno moramo iskati. Naši pametni telefoni se razvijajo v smeri, ki nam skuša posredovati informacije o stvareh, ki nas zanimajo, na karseda prijazen in za uporabnika nezahteven, če že ne neopazen način. Prav tako želimo, da naši pametni telefoni zmorejo čim več storitev, ki so nam jih nekoč nudile specializirane naprave.

Poglavje 2

Obstoječe rešitve

Inspiracijo črpamo iz spletnega portala SPIN, ki zbira prijave o izrednih dogodkih na številki 112. Spletni portal je pri objavljanju prijav zelo agilen, podatki o prijavljenih dogodkih so objavljeni zelo hitro. Prav tako ponuja naročanje na obveščanje, a preko vira RSS, in ne preko mobilne aplikacije. Na trgu je veliko aplikacij, ki obveščajo ljudi o prijetnejših dogodkih, takšnih, ki obveščajo o izrednih dogodkih, pa smo zasledili bolj malo. Poleg spletnega portala SPIN pa smo izvedeli še za mobilno aplikacijo Fire Alert in spletni portal Intervencije.net.

2.1 Mobilna aplikacija Fire Alert

Iz opisa aplikacije v trgovini Google Play izvemo, da se aplikacija prav tako obnaša kot pozivnik. Omogoča nam, da na prejeta SMS, MMS, elektronska sporočila in klice nastavimo prožilce. Prožilcem nastavimo ključne besede, pošiljatelje oziroma klicatelje, ter način na katerega želimo, da nas aplikacija o njih obvesti. Ob sproženem prožilcu se tako naš telefon obnaša kot pozivnik po prej nastavljenih parametrih. Nastavimo lahko obvestila v vrstici za obvestila, lastnosti zvonjenja, prižig ekrana ali bliskavice in še več. Aplikacija nam torej ponuja veliko možnosti za sam način obveščanja, a zahteva, da smo o dogodkih že nekako obveščeni. Avtor Sven Hörnchen v opisu aplikacije tudi poudari, da je le-ta brezplačna in brez oglasov, namenjena izključno reševanju življenj in materialnih dobrin.

2.2 Spletni portal Intervencije.net

Portal omogoča posredovanje obvestil s pozivnika na mobilni telefon s pomočjo sporočila SMS. S storitvijo želijo odpraviti časovne zakasnitve in posledice morebitnih slabih radijskih signalov pozivnikov, sporočilo pa naj bi prispelo v roku desetih sekund od prejetja obvestila na pozivnik in naj ne bi imelo zakasnitve zaradi sočasnega pošiljanja več sporočil. Dodatna funkcionalnost portala, ki je bila tudi povod za izdelavo sistema pa je, da se pozvani lahko na intervencije odzovejo s klicem na določene številke. Vodja intervencije lahko tako na posebnem seznamu preveri, kdo na intervencijo pride in kdo ne. V predstavitvi sistema so zapisali opozorilo, da se sistem uporablja le kot dopolnilni sistem, in nikakor ne nadomešča sistema pozivnikov, ter da so imetniki pozivnikov le-te dolžni uporabljati. Storitev je dostopna društvom in drugim intervencijskim enotam, ki imajo sklenjeno pogodbo za uporabo portala. Storitev je plačljiva. V mnenjih uporabnikov pa lahko zasledimo, da storitev dobro dopolni prej opisana aplikacija Fire Alert.

Poglavje 3

Uporabljene tehnologije

Sistem je v grobem sestavljen po klasični strežnik-odjemalec arhitekturi. Sestavni deli posameznih delov sistema temeljijo na uveljavljenih odprtokodnih tehnologijah.

3.1 Strežniški del

Strežniški del sistema, kot že samo ime pove, teče na strežniku. Njegova vloga v sistemih ponavadi je, da streže odjemalcem, procesira njihove zahteve in skrbi za shranjevanje podatkov. V okviru strežniškega dela smo uporabili tehnologije, ki so opisane v nadaljevanju tega podpoglavja.

3.1.1 Java

Java je programski jezik, ki temelji na razredih, je objektno orientiran in posebej zasnovan tako, da ima kar se da malo odvisnosti pri izvajanju. Njen namen je, da razvijalci pišejo kodo le enkrat, poganjajo pa jo lahko kjerkoli, kar pomeni, da kode, ki se izvaja na eni platformi, ni potrebno ponovno prevajati za izvajanje na drugi. Javanske aplikacije so tipično prevedene v datoteke s končnico .class, ki se lahko izvajajo na vsakem Javanskem virtualnem stroju (JVM), neodvisno od računalniške arhitekture.

Java je eden najpopularnejših programskih jezikov, predvsem pri razvoju spletnih aplikacij, kjer jo uporablja 9 milijonov razvijalcev [1, 2].

Začetki Jave segajo v leto 1995, v času pisanja diplomskega dela pa je za razvijalce na voljo že deseta verzija, poimenovana Java SE 8, Java pa je pod okriljem Oracle Corporation.

3.1.2 Java EE

Je poslovna različica platforme Java, ki ponuja vmesnik za programiranje aplikacij (API) in okolje za izvajanje poslovnih aplikacij. Java EE razširja platformo Java (Java SE) [3] in med drugim ponuja API za objektno-relacijsko preslikovanje, za porazdeljene in večnivojske arhitekture ter spletne storitve. Platforma inkorporira zasnovo in bazira na modularnosti komponent, ki tečejo na aplikacijskem strežniku. Platforma poudarja konvencijo nad konfigiracijo in uporabo anotacij za konfiguracijo [4].

3.1.3 Enterprise JavaBeans

Enterprise JavaBeans (EJB) je v grobem platforma za gradnjo prenosljivih, ponovno uporabnih in razširljivih poslovnih aplikacij v jeziku Java. Iz perspektive razvijalca je EJB kos javanske kode, ki se izvaja v vsebovalniku EJB [6]. Vsebovalnike za izvajanje zrn EJB nudijo razni aplikacijski strežniki.

3.1.4 JBoss aplikacijski strežnik

Aplikacijski strežnik JBoss je odprtokoden projekt, ki je trenutno pod okriljem skupnosti Red Hat. Predstavlja implementacijo Java EE specifikacij in nudi polno podporo za razvoj in izvajanje aplikacij Java EE. Napisan je v programskem jeziku Java. Njegovi začetki segajo v leto 1999, tako kot sam začetek Jave EE. Skozi ta čas je doživel številne izbolšave, v času pisanja diplomskega dela pa je na voljo že 8. verzija. Projekt je sam po sebi odprtokoden, Red Hat pa zaračunava podporo.

Strežniški del Sistema za obveščanje o izrednih dogodkih teče na omenjenem strežniku, katerega smo izbrali zaradi odprtokodnosti, enostavnosti uporabe in celovite implementacije specifikacij Jave EE.

3.1.5 Hibernate

Hibernate je odprtokodna knjižnica za objektno-relacijsko preslikovanje (ORM). Objektno-relacijsko preslikovanje je avtomatizirano in transparentno persistiranje javanskih objektov v tabele relacijske podatkovne baze z uporabo meta podatkov, ki opisujejo preslikave med objekti in podatkovno bazo [5].

Hibernate, ne samo da poskrbi za preslikovanje javanskih razredov v tabele podatkovnih baz (in iz javanskih podatkovnih tipov v SQL podatkovne tipe), ampak tudi ponuja pristope za podatkovne poizvedbe in pridobivanje podatkov, ter tako lahko znatno skrajša čas razvoja, ki bi bil drugače porabljen za ročno rokovanje s podatki v SQL in JDBC [8].

V začetku je Hibernate pod svoje okrilje vzel JBoss, oba pa sta danes del Red Hata. Predstavlja implementacijo JPA specifikacij.

3.1.6 MySQL

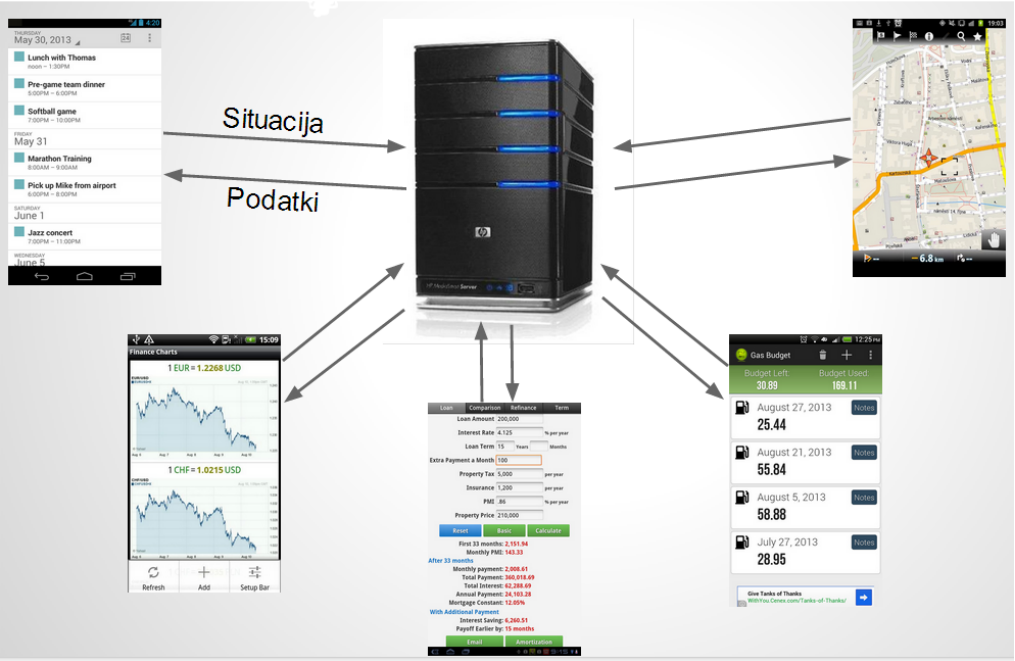
MySQL je odprtokodni sistem za upravljanje z relacijskimi podatkovnimi bazami. Poleg podatkovne baze ponuja tudi orodja za delo s podatki v podatkovni bazi. MySQL je napisan v programskem jeziku C in C++. Za delo s podatkovno bazo SQL v Java aplikaciji je potreben JDBC (Java Database Connectivity) API [5]. Za povezavo z bazo, JDBC zahteva gonilnik. Gonilnik JDBC je komponenta, ki omogoča Java aplikaciji interakcijo s podatkovno bazo.

3.1.7 Kontekstni strežnik

Kontekstni strežnik pri posredovanju podatkov napravam upošteva tudi njihov kontekst in tako poskuša dostaviti čim bolj relevantne podatke.

Ideja kontekstnega strežnika je ta, da uporabnikom naprav servira relevantne podatke, ki se izbirajo glede na njihov kontekst v širšem smislu. Kontekst je sestavljen iz fizičnega, psihičnega, socialnega, informacijskega in tehnološkega. V skladu s skupnim kontekstom se uporabniku posredujejo različni podatki in obvestila.

Strežnik od naprave redno dobiva kontekstne podatke in podatke o uporabi naprave, jih obogati, shrani in procesira poslovna pravila ter napravi vrne relevantne podatke, če takšni obstajajo [9], 3.1. Kontekstne podatke pridobiva na podlagi prej zastavljenih in podprtih poslovnih pravil.



Slika 3.1: Procesiranje konteksta [9].

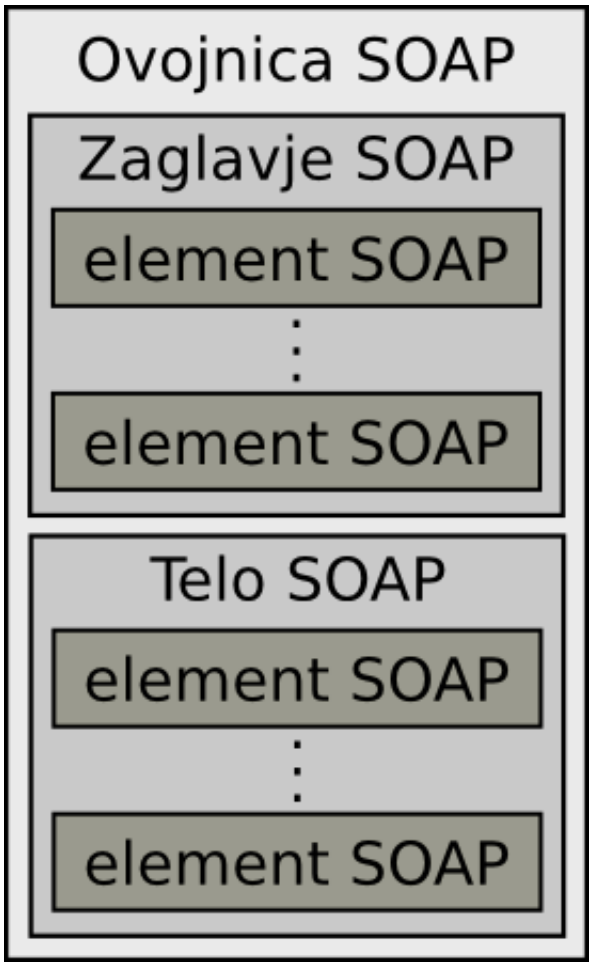
3.1.8 Spletne storitve

Spletna storitev je način komunikacije med dvema napravama po omrežju. Najpogostejši vrsti spletnih storitev sta SOAP in REST.

3.1.8.1 Spletne storitve SOAP

Simple Object Access Protocol (SOAP) je platformno neodvisni protokol, ki uporablja razširljiv označevalni jezik (XML) za oddaljene klice procedur, tipično preko protokola HTTP. Vsaka zahteva in odgovor sta zapakirana v sporočilo SOAP-XML. Sporočilo nosi informacije, ki so potrebne, da ga spletna storitev sprocesira. Sporočila so zaradi jezika XML razumljiva računalniku in ljudem, ter so platformno neodvisna [7].

Spletna storitev je opisana z datoteko WSDL. Sporočilo SOAP je sestavljeno iz ovojnice SOAP, katero sestavljata zaglavje in telo SOAP 3.2.



Slika 3.2: Struktura ovojnice SOAP na podlagi [23].

3.1.8.2 Spletne storitve REST

Representational State Transfer (REST) je arhitekturni stil za razvoj spletnih storitev. Takšnim spletnim storitvam pogosto pravimo RESTful spletne storitve. Vsaka metoda RESTful spletne storitve je definirana z enoličnim naslovom URL. Za razliko od spletnih storitev SOAP, zahteve in odgovori niso oviti v ovojnice. REST prav tako ni omejen na vračanje podatkov v formatu XML [7].

Tudi REST je platformno neodvisen in lahko povezuje različne aplikacije.

3.1.8.2.1 RESTEasy je projekt oddelka JBoss podjetja Red Hat in je odprtokodna certificirana implementacija specifikacije JAX-RS [21].

3.2 Mobilna aplikacija

Mobilna aplikacija je računalniški program, zasnovan za izvajanje na pametnih telefonih, tabličnih računalnikih in mobilnih napravah [10].

V okviru mobilne aplikacije smo uporabili tehnologije, ki so opisane v nadaljevanju tega podpoglavja.

3.2.1 Android

Android je programsko okolje, zasnovano za mobilne naprave. Sestavljen je iz operacijskega sistema, ki bazira na kernelu Linux, bogatega uporabniškega vmesnika, aplikacij za končne uporabnike, knjižnic, multimedijske podpore in še mnogo več. Osnovne komponente operacijskega sistema so pisane v jeziku C in C++, uporabniške in vgrajene aplikacije pa so pisane v Javi. Ena izmed prednosti platforme Android je, da se uporabniške aplikacije, razvite z Android SDK, ne razlikujejo od vgrajenih, kar pomeni, da lahko pišemo zelo močne aplikacije, s katerimi lahko koristimo vire, ki jih naprava ponuja. Ena izmed opaznejših prednosti Androida je tudi, da ga zaradi njegove odprtokodnosti lahko dopolnjuje širša razvijalska skupnost [11].

Android je trenutno najpopularnejša mobilna platforma na svetu, teče na stotinah milijonov naprav v več kot 190 državah sveta [12].

K razvoju Androida je največ prispeval Google, ki je od leta 2005 tudi njegov lastnik.

3.2.2 Android Manifest

AndroidManifest.xml datoteka je relativno majhen, a zelo pomemben del Android aplikacije. Datoteka nosi informacije o aplikaciji, njenih aktivostih, minimalnem in ciljnem nivoju SDK, pravicah, ki jih aplikacija zahteva, in nastavitvah za izvajanje aplikacije.

3.2.3 Google Play Services

Google Play Services je zbirka orodij za razvoj programske opreme (SDK) in vmesnik za programiranje aplikacij (API). Čeprav deluje kot proces na ravni sistema, se lahko samodejno posodablja preko trgovine Google Play in se samodejno namesti na katero koli napravo Android z operacijskim sistemom Android 2.2 ali novejše. Od njega so odvisne vse Googlove Android aplikacije [13].

Predstavlja enostaven dostop do Googlovih storitev in je tesno povezan z operacijskim sistemom Android. Ponuja knjižnice za odjemalce, ki so enostavne za uporabo, in ponuja storitve, s katerimi lažje in hitreje implementiramo željene funkcionalnosti. Predstavljen je z diagramom 3.3.

S pristopom pakiranja v APK in distribuiranja preko trgovine Google Play so dosegli neodvisnost izboljšav in nadgradenj od samih verzij Androida. Tako so popravki napak in posodobitve lahko bolj pogosti in dostopni večjem številu naprav.

3.2.4 Google Cloud Messaging

Google Cloud Messaging (GCM) je storitev v sklopu prej opisanih storitev Google Play. GCM razvijalcem omogoča pošiljanje podatkov iz aplikacijskih strežnikov na aplikacije Android naprav ter povratna sporočila uporabnikovih naprav nazaj v oblak. Sporočila lahko uporabnika enostavno obvestijo o tem, da ga na strežniku čakajo novi podatki, ali pa že sama vsebujejo do 4kb podatkov v sporočilu.

Da lahko aplikacija Android prejme sporočilo, ni potrebno, da je le-ta v izvajanju. Za prejemanje sporočil je dovolj implementacija ustreznega sprejemnika in

Slika 3.3: Google Play Services APK na uporabnikovih napravah redno prejema posodobitve za nove APIje, funkcionalnosti in popravke [13].

specifikacija pravic v manifestu aplikacije. GCM aplikaciji Android dostavi surovo sporočilo, ta pa ima vso kontrolo nad tem, kaj z njim narediti. Lahko na primer uporabniku prikaže sporočilo, poseben uporabniški vmesnik, ali pa po tihem sinhronizira podatke.

3.2.4.1 Osnovni koncepti

Ključne pojme in koncepte GCM lahko delimo na dve skupini, in sicer na komponente in pravice (angl. credentials) [14]. V nadaljevanju podpoglavja sledi podrobnejši opis osnovnih pojmov in konceptov.

3.2.4.1.1 Odjemalska aplikacija Odjemalska aplikacija mora imeti omogočen GCM, teči pa mora na napravi, ki ima nameščeno trgovino Google Play in Android verzije 2.2 ali novejše.

3.2.4.1.2 Aplikacijski strežnik Aplikacijski stežnik je strežnik, s pomočjo katerega lahko razvijalec preko povezovalnega strežnika GCM pošlje podatke odjemalski aplikaciji Android.

3.2.4.1.3 Povezovalni strežnik GCM Povezovalne strežnike GCM zagotavlja Google, njihova naloga v grobem pa je, da sporočila z drugih strežnikov posredujejo odjemalskim aplikacijam.

3.2.4.1.4 Pravice (Credentials) Preden lahko začnemo, moramo v Google-konzoli za razvijalce (angl. Google Developers Console) ustvariti projekt Google API. S tem postopkom pridobimo kode in pravice za uporabo storitve GCM.

3.2.4.1.4.1 Pošiljateljev ključ (Sender ID) Z registracijo projekta v Google-konzoli za razvijalce pridobimo številko projekta. Ta številka predstavlja naš strežnik v procesu registracije naprave.

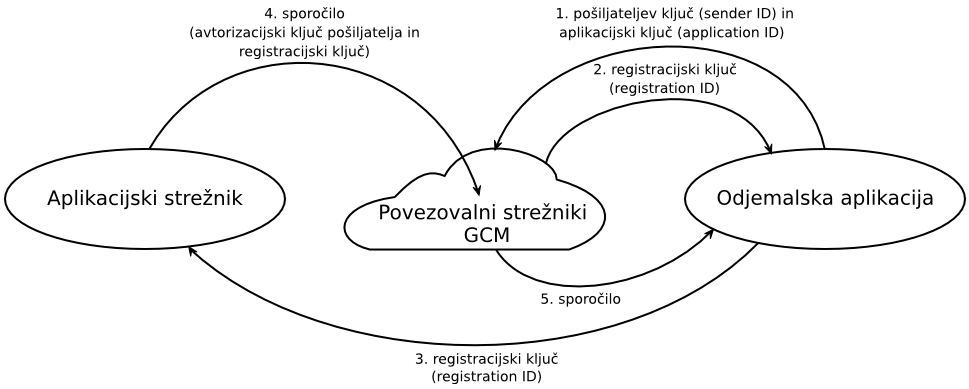
3.2.4.1.4.2 Aplikacijski ključ (Application ID) Identifikacija predstavlja aplikacijo Android, ki se naroča na prejemanje sporočil. Identifikacija se določi na podlagi imena paketa iz manifesta aplikacije in zagotovi, da so sporočila namenjena pravi aplikaciji Android.

3.2.4.1.4.3 Registracijski ključ (Registration ID) Identifikacija enolično predstavlja aplikacijo na napravi Android. Aplikaciji je dodeljena s strani strežnikov GCM, ta pa jo posreduje aplikacijskem strežniku, ki jo bo uporabil za naslavljanje naprave ob pošiljanju sporočil.

3.2.4.1.4.4 Uporabniški račun Google (Google User Account) Če na mobilni napravi teče verzija Androida, ki je starejša od 4.0.4, mora le-ta za delovanje GCM vsebovati vsaj en Google-račun.

3.2.4.1.4.5 Avtorizacijski žeton pošiljatelja (Sender Auth Token) Je ključ (angl. API key), ki daje aplikacijskem strežniku avtoriziran dostop do Google-storitev in je vključen v glavo zahtev POST, ki pošiljajo sporočila. Tudi tega pridobimo v Google-konzoli za razvijalce, kjer pa tudi določimo, s katerih naslovov IP bo naš aplikacijski strežnik lahko pošiljal sporočila.

Implementacija GCM torej vključuje aplikacijski strežnik, povezovalne strežnike GCM in odjemalske aplikacije na napravah Android z omogočenim GCM.



Slika 3.4: Sekvenčni diagram naročanja in prejemanja sporočila na podlagi [15].

Android aplikacija se mora na prejemanje sporočil najprej naročiti. Ko povezovalni strežnik GCM sprejme sporočilo aplikacijskega strežnika, ga posreduje Android aplikaciji z omogočenim GCM 3.4.

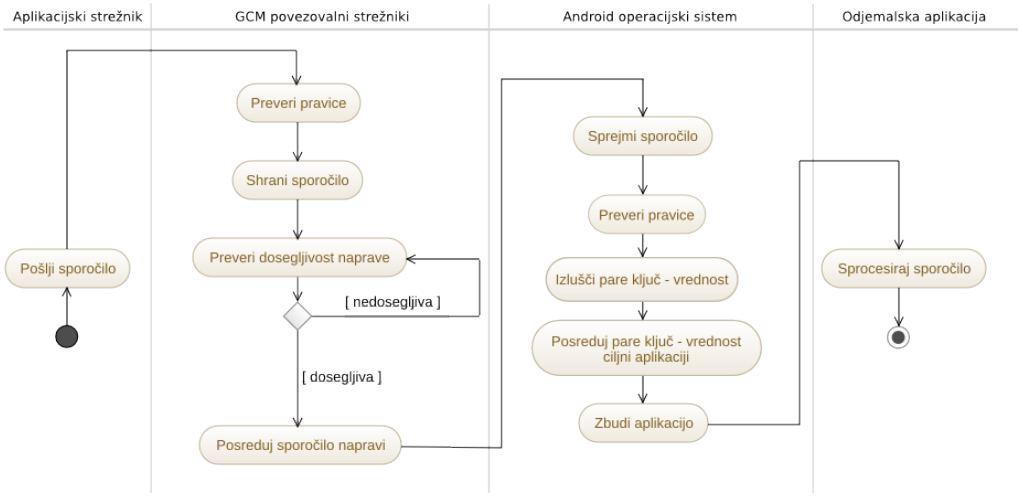
3.2.4.2 Pošiljanje in prejemanje sporočila

Ob pošiljanju sporočila se zgodijo naslednji dogodki [16]:

1. Aplikacijski strežnik pošlje sporočilo strežniku GCM.
2. Google sprejme sporočilo in ga shrani za primer, če je naprava nedosegljiva.
3. Ko je naprava dosegljiva, ji Google pošlje sporočilo.
4. Sistem na napravi posreduje sporočilo določeni aplikaciji Android preko Intent broadcasta z ustreznimi pravicami, kar omogoča, da sporočilo dobi samo ciljna aplikacija Android. Ta akcija zbudi aplikacijo. Aplikaciji ni potrebno biti v teku, da prejme sporočilo.
5. Aplikacija Android sprocesira sporočilo.

Tem procesu rečemo tudi push ali potis, sporočilu pa posledično potisno sporočilo.

Ko aplikacija Android, nameščena na mobilni napravi prejme sporočilo, se zgodijo naslednji dogodki [17]:



Slika 3.5: Diagram aktivnosti UML pošiljanja in prejemanja sporočila na podlagi [16].

1. Sistem prejme surovo sporočilo in iz njega izlušči pare ključ-vrednost, če ti obstajajo.

2. Sistem posreduje pare ključ-vrednost ciljni aplikaciji Android v Intent, definiran v manifestu s ključem "com.google.android.c2dm.intent.RECEIVE", kot set dodatkov (angl. Extras).

3. Aplikacija Android izlušči surove podatke iz omenjenega Intenta preko ključa in sprocesira podatke.

Procesa sta predstavljena z diagramom aktivnosti UML pošiljanja in prejemanja sporočila 3.5.

3.2.5 GPS

GPS je kratica za Global Positioning System oziroma sistem za globalno pozicioniranje. Sistem je sestavljen iz navigacijskih satelitov v vesolju, ki zagotavljajo informacijo o lokaciji in času v vseh vremenskih razmerah, kjer koli na, oziroma blizu zemlje, kjer je lokacija vidna vsaj štirim ali več satelitom GPS [18].

Uporablja se v različne namene s pomočjo širokega nabora različnih naprav, ki ga podpirajo in je brezplačen. S pocenitvijo in pomanjševanjem tehnologije GPS in sprejemnikov, pa so ti lahko postali nepogrešljiv del mobilnih telefonov.

3.2.5.1 Google Location API

Google ponuja v sklopu storitev Google Play vmesnik za programiranje aplikacij (API) za enostavno razvijanje aplikacij, ki uporabljajo lokacijo. Tako kot pri vseh vmesnikih so tudi tukaj podrobnosti tehnologije za razvijalca skrite, kar omogoča, da lahko razvijalec več časa posveti sami aplikaciji, ne da bi se moral osredotočati na podrobnosti tehnologije lociranja. Google storitev lokacije inteligentno upravlja s tehnologijo lociranja in nudi najboljšo lokacijo glede na razvijalčeve potrebe. Omogoča nastavljivo natančnost lociranja, hiter dostop do lokacije in učinkovito rabo baterije. Med drugim omogoča postavljanje geografskih meja in spremljanje gibanja uporabnika glede nanje ter prepoznavanje načina premikanja uporabnika - mirovanje, hoja, kolesarjenje, avtomobil [19].

V nadaljevanju sta predstavljena dva razreda, ki smo ju uporabili za delo z lokacijo pri upoštevanju uporabnikovega fizičnega konteksta.

3.2.5.1.1 LocationManager *LocationManager* je razred v paketu *android.location*, ki med drugim omogoča, da lokacijam nastavimo opozorilo bližine (angl. proximity alert). Ko se naprava znajde znotraj podanega radija lokacije z nastavljenim opozorilom bližine, ta sproži čakajoč intent (PendingIntent), ki je prav tako podan kot parameter metode *addProximityAlert* [20].

3.2.5.1.2 Geocoder *Geocoder* je prav tako del paketa *android.location*, ki pa med drugim omogoča pridobivanje lokacije na podlagi naslova.

3.3 SPIN

SPIN je informacijski sistem poročanja o intervencijah in nesrečah. Razvilo ga je podjetje LOGOS.SI d.o.o za potrebe Uprave RS za zaščito in reševanje. V sistem poročanja so vključeni vsi centri za obveščanje (112) in ostale enote za zaščito in reševanje v Sloveniji. Sistem aktivno uporablja več kot 3500 uporabnikov. Poleg

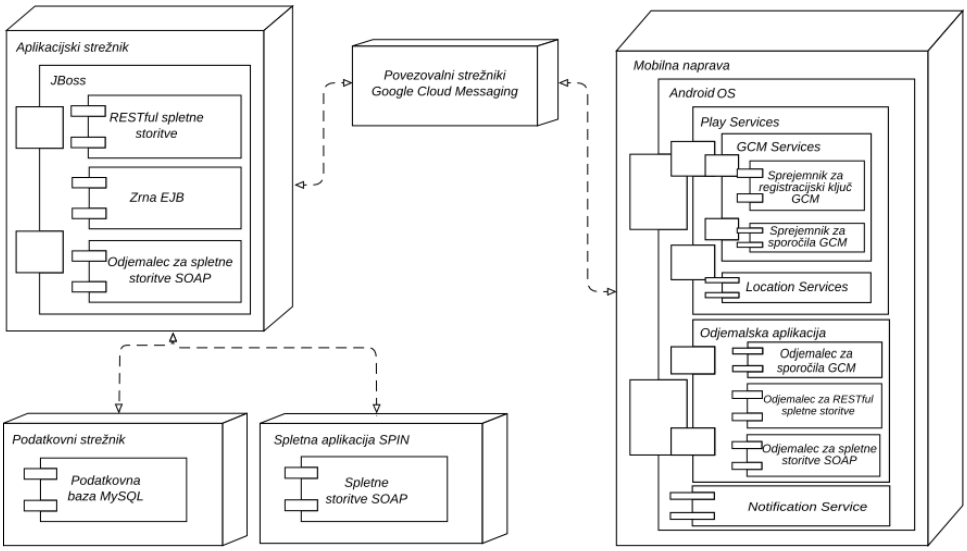
tega pa so nekateri deli sistema, preko spletnih strani Uprave RS za zaščito in reševanje, neposredno dostopni neregistriranim uporabnikom [22].

Kot neregistrirani uporabnik lahko vidimo grafični prikaz trenutnega stanja lokacij in podatkov o nesrečah v Sloveniji, prikaz dogodkov in poročil o dogodkih. Prav tako pa se neregistrirani uporabnik lahko naroči na obveščanje o dogodkih z viri RSS. To stori s pomočjo spletnega obrazca, kjer označi področja, ki ga zanimajo. Področja obsegajo območja Slovenije, vrste dogodkov (naravne nesreče, požari, itd.) in aktivirane enote (gasilci, helikopterske enote, itd.). Uporabnik lahko tudi izbere, ali ga naj obvestijo o dogodku ob prvi prijavi, ali pa ko je dogodek preverjen. Spletna aplikacija uporabniku glede na izbrane parametre vrne povezavo do vira RSS.

Poglavje 4

Opis rešitve

V nadaljevanju bomo podrobneje opisali našo rešitev, njene sestavne dele in delovanje. Opis bo razdeljen na opis strežniškega dela in mobilne aplikacije. Diagram 4.1 prikazuje sestavne dele sistema in njihove komponente, ki so pomembne za obveščanje o izrednih dogodkih.



Slika 4.1: Postavitveno komponentni diagram UML prikazuje sestavne komponente prototipa sistema, ki so pomembne za obveščanje o izrednih dogodkih.

4.1 Strežniški del

Glavna vloga strežniškega dela je, da neprestano poizveduje po novih izrednih dogodkih, prijavljenih na spletnem portalu SPIN, in v primeru izrednega dogodka o tem obvesti naročene uporabnike. Ključna naloga je torej tudi, da vodi seznam naročenih uporabnikov in njihovih nastavitev ter komunicira s povezovalnimi strežniki GCM.

4.1.1 Podatkovna baza

Podatkovna baza je sestavljena iz tabel, ki nosijo podatke, potrebne za pošiljanje sporočil GCM in tabel, ki nosijo informacije o uporabniških nastavitvah obveščanja.

Za delo s podatkovno bazo smo uporabili odprtokodno knjižnico za objektno-relacijsko preslikovanje Hibernate. Ta v grobem na podlagi anotacij in konfiguracijskih datotek XML preslika javanske razrede v tabele, instance objektov pa v zapise v teh tabelah. Ponuja pa tudi poizvedovalen jezik HQL, ki omogoča, da lahko poizvedujemo na podatkovnih bazah različnih ponudnikov, brez prirejanja poizvedb.

Hibernate poskrbi za preslikavo relacij kot je ena-mnogo (angl. one-to-many, tudi 1-*) v povezovalne tabele, ki nosijo ključe povezanih entitet. Teh tabel ne bomo posebej opisovali, ampak bomo na entitete in atribute posameznih entitet pogledali objektno. Podroben opis razredov in objektno-relacijskega preslikovanja sledi v nadaljevanju podpoglavja.

4.1.2 Objektno-relacijsko preslikovanje

Paket *entities* nosi objektne predstavitve tabel. Vsi razredi, ki predstavljajo tabele, razširjajo abstrakten razred *AbstractEntity*, ki implementira vmesnik *Serializable* in nosi atribut, skupen vsem tabelam - ključ.

Izvorna koda 4.1: Abstraktni razred *AbstractEntity* implementira vmesnik *Serializable* in služi kot osnova vsem persistentnim entitetam, medtem ko sam nima relacijske predstavitve.

```
@MappedSuperclass
public abstract class AbstractEntity implements Serializable {
    private Long id;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }
}
```

Razredi, pomembni za obveščanje s pomočjo tehnologije Google Cloud Messaging, so: *AndroidApplication*, *GcmMessage*, *GcmMessageData* in *GcmRegistrationId*.

Entiteta *AndroidApplication* predstavlja aplikacijo Android. Objektno gledano so njeni atributi naslednji:

- **String projectId** - project id aplikacije Android
- **String projectNumber** - project number aplikacije Android, ki ga pridobimo, ko registriramo projekt v Google-konzoli za razvijalce
- **List<GcmMessage> gcmMessages** - seznam sporočil GCM
- **List<GcmRegistrationId> registrationIds** - seznam registracijskih ključev GCM

Entiteta *GcmMessage* nosi atribute sporočila Googe Cloud Messaging:

- **String collapseKey** - ključ, pod katerim se grupirajo obvestila
- **Integer timeToLive** - življenjska doba sporočila
- **Boolean delayWhileIdle** - atribut, ki pove, ali naj obvestilo speč telefon zbudi, ali naj počaka
- **List<GcmMessageData> gcmMessageDataList** - seznam podatkov sporočila

GcmMessageData nosi podrobnosti *GcmMessage* v obliki, kot jih prejme aplikacija Android:

- **String intentExtraName** - ime oziroma ključ podatka, kot ga dobi aplikacija Android
- **String intentExtraValue** - vrednost za ključ oziroma podatek

GcmRegistrationId predstavlja registracijski ključ, ki enolično predstavlja napravo in na njej nameščeno aplikacijo:

- **String registrationId** - registracijski ključ
- **Subscription subscription** - objekt, ki služi kot povezava med registracijskim ključem in uporabniškimi nastavitvami obveščanja

Entiteta *Subscription* nosi uporabniške nastavitve za obveščanje in je povezana z registracijskim ključem s povezavo ena-ena (angl. one-to-one, tudi 1-1):

- **boolean unchecked** - atribut pove, ali naj uporabnika obvestimo tudi o nepreverjenih dogodkih
- **boolean proximityEnhanced** - atribut pove, ali uporabnik želi dodatno obvestilo, ko se nahaja v neposredni bližini izrednega dogodka
- **List<SubLocation> locations** - seznam lokacij, na katere je uporabnik naročen
- **List<SubType> types** - seznam tipov dogodkov, na katere je uporabnik naročen
- **List<Unit> units** - seznam enot, ob aktivaciji katerih je uporabnik naročen na obvestilo

Entiteta *SubLocation* je povezana z entiteto *Location* s povezavo mnogo-ena (angl. many-to-one, tudi *-1). Entiteta *Location* je torej sestavljena iz seznama entitet *SubLocation*, obema pa so preostali atributi skupni, podedujeta jih od entitete *CodeName*, ki služi kot šifrant:

- **int code** - šifra
- **String name** - ime

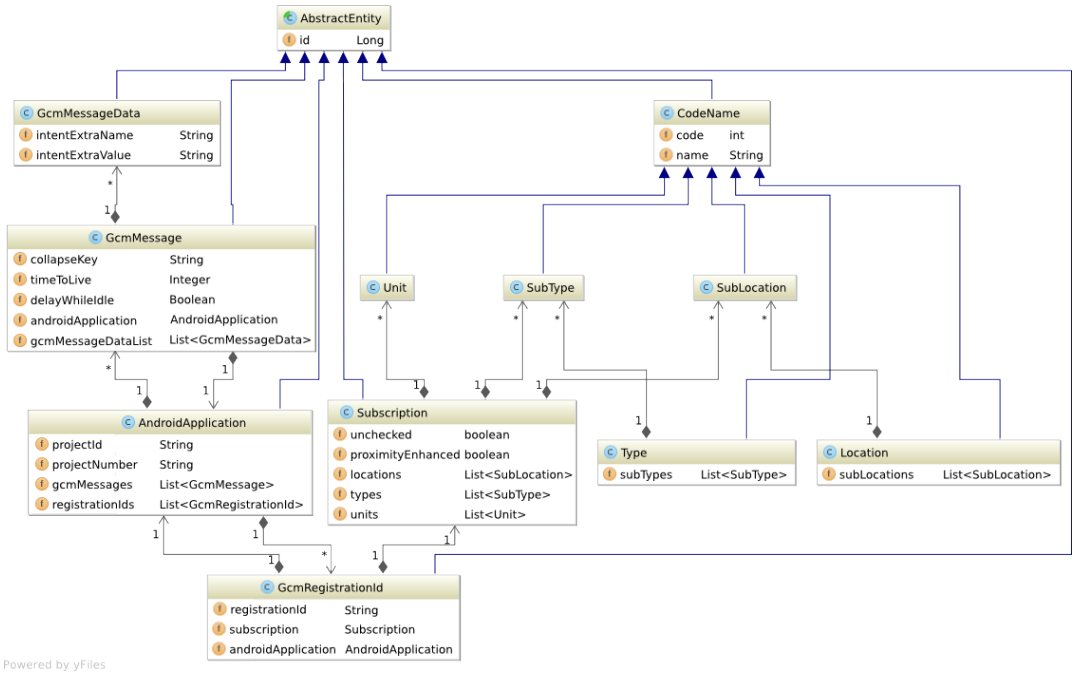
Entiteta *SubType* je povezana z entiteto *Type* s povezavo mnogo-ena. Entiteta *Type* je torej sestavljena iz seznama entitet *SubType*, obema pa so preostali atributi skupni, podedujeta jih prav tako od entitete *CodeName*.

Entiteta *Unit* vsebuje samo atribute, podedovane od entitete *CodeName*, in tako predstavlja enostaven šifrant.

Izvorna koda 4.2: Razred *CodeName* služi kot osnova vsem šifrantom, medtem ko sam nima relacijske predstavitve.

```
@MappedSuperclass
public class CodeName extends AbstractEntity {
    private int code;
    private String name;

    public CodeName() {
    }

    public CodeName(int code, String name) {
        this.code = code;
        this.name = name;
    }

    public int getCode() {
        return code;
    }

    public void setCode(int code) {
        this.code = code;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

Opisani razredi so predstavljeni z diagramom 4.2.



Slika 4.2: Diagram razredov objektne predstavitve podatkovne baze.

Šifranti so potrebni za poizvedovanje po dogodkih na spletnem portalu SPIN. Za vstavljanje šifrantov v podatkovno bazo iz datoteke CSV so bili napisani posebni razčlenjevalniki (angl. parser).

4.1.3 Registracija naprave

Strežniški del za registracijo naprav ponuja posebne spletne storitve REST. Te se nahajajo v *webservices* paketu *web* modula spletne aplikacije. Vse spletne storitve so registrirane na naslovu */rest*, storitve, potrebne za registracijo naprave pa se nahajajo na naslovu */rest/gcm*. Na tem naslovu lahko dostopamo do dveh spletnih storitev */rest/gcm/registerDevice* in */rest/gcm/registerParameters*.

Spletne storitve so implementirane s pomočjo RESTEasy implementacije specifikacije JAX-RS.

Spletna storitev */registerDevice* pričakuje naslednje parametre:

- **String projectNumber** - številka projekta, s pomočjo katere lahko registracijski ključ dodeli ustrezni Android aplikaciji

- **String registrationId** - registracijski ključ, ki enolično predstavlja naročnika na obveščanje

Izvorna koda 4.3: Spletna storitev *registerDevice* je implementirana s pomočjo RESTEasy implementacije specifikacije JAX-RS.

```
@Path("/gcm")
public class GcmServices {
    @POST @Path("/registerDevice")
    public void registerDevice(@FormParam("projectNumber") String projectNumber, @FormParam("
         registrationId") String registrationId) {
        Session session = null;
        try {
            session = HibernateUtil.getSessionFactory().getCurrentSession();
            session.beginTransaction();

            GcmAgent gcmAgent = new GcmAgent(session);
            AndroidApplication androidApplication = gcmAgent.getAndroidApplication(GcmAgent.
                 GCM_PROJECT_ID, projectNumber);

            if (androidApplication == null) {
                androidApplication = new AndroidApplication(GcmAgent.GCM_PROJECT_ID,
                     projectNumber);
                session.save(androidApplication);
            }

            gcmAgent.addGcmRegistrationId(androidApplication, registrationId);
            session.update(androidApplication);
        } catch (Exception e) {
            log.error("An exception occurred in \"Register device\" web service.");
            if (session != null) {
                session.getTransaction().rollback();
            }
        }
    }
    ...
```

4.1.4 Upoštevanje konteksta

Sistem za obveščanje pošilja obvestila uporabnikom na podlagi njihovih preferenc. Uporabnik lahko v nastavitvah določi, ali želi, da je obveščen o dogodku, še preden je preverjen, kateri tipi dogodkov ga zanimajo, katere lokacije in katere intervencijske enote.

Uporabnik lahko tudi omogoči nastavitev, ki bo upoštevala njegovo lokacijo in ga v primeru, da se nahaja v neposredni bližini izrednega dogodka o tem tudi dodatno obvestila. Na ta način smo upoštevali eno najpomembnejših informacij o uporabnikovem fizičnem kontekstu, njegovo lokacijo.

4.1.5 Poizvedovanje po posodobitvah

Ena izmed ključnih nalog strežniškega dela je, da neprestano poizveduje po posodobitvah na spletnem portalu SPIN. Mehanizem, ki skrbi za neprestano proženje poizvedovanj ob želenih trenutkih, je posebna anotacija *Schedule*, ki je na voljo kot del EJB 3.1. Anotaciji z atributom povemo, kdaj naj sproži metodo, ki jo anotira. Anotacija prav tako sprejme parameter, ki pove, ali naj bo prožilec persistenten ter tako shranjen v posebni mapi na strežniku, kjer bo preživel morebitno ustavitev in ponoven zagon strežnika.

Izvorna koda 4.4: Anotacija *Schedule* sproži anotirano metodo vsako polno minuto vsake ure.

```
@Singleton
public class GBUpdatesCheckerEJB {
    private long lastUpdateId = 0;
    private long lastUncheckedUpdateId = 0;
    @Resource SessionContext sessionContext;

    @Schedule(minute = "*/1", hour = "*")
    public void automaticTimeout() {

        log.debug("Checking for updates. Last update id: " + lastUpdateId);
        log.debug("Checking for unchecked updates. Last unchecked update id: " +
             lastUncheckedUpdateId);

        checkForUncheckedUpdates();
        checkForUpdates();
    }
    ...
```

Poizvedovanje na spletnem portalu SPIN poteka preko posebne spletne storitve, ki jo je podjetje LOGOS.SI d.o.o., ki je ustvarilo sam portal SPIN, pripravilo posebej za nas. Opis spletne storitve s primeri uporabe lahko najdemo na naslovu

`http://spin.sos112.si/spin2ws/SPIN2Service.asmx?op=OD`. Kot lahko razberemo iz opisa spletne soritve, ta prejme vse parametre, ki jih uporabnik lahko izbere na spletnem portalu SPIN.

4.1.6 Obveščanje naprav

V primeru pojava novega izrednega dogodka strežniški del preveri svoje naročnike in njihove nastavitve obveščanja, ter jih, če dogodek ustreza njihovim nastavitvam, o njem obvesti. To stori s pomočjo tehnologije Google Cloud Messaging. Najprej sporočila sestavi v skladu s predpisano obliko in jim nastavi predpisane parametre.

Parametri, ki jih nastavimo, so naslednji:

- **String collapseKey** - ključ, pod katerim se na napravah grupirajo obvestila, v našem primeru "SOID".
- **Integer timeToLive** - življenjska doba sporočila pove, koliko časa bodo povezovalni strežniki hranili sporočilo in ga poskušali dostaviti uporabniku. V našem primeru smo atribut nastavili na 3600 sekund. Na ta način povezovalnim strežnikom povemo, naj 3600 sekund poskušajo uporabnika obvestiti o dogodku, tudi če je ta v trenutku, ko prvič poskusijo dostaviti obvestilo, zaradi takšnih ali drugačnih razlogov nedosegljiv.
- **Boolean delayWhileIdle** - atribut, ki pove, ali naj obvestilo speč telefon zbudi (false), ali naj počaka (true), smo nastavili na false.

Samo sporočilo služi kot opomnik uporabniku mobilne aplikacije, naj osveži podatke o dogodkih. Takšnem sporočilu rečemo tudi sporočilo send-to-sync. V primeru, da je uporabnik naročil upoštevanje lokacije, pa sporočilo nosi tudi informacijo o lokaciji novega dogodka, tako da lahko naprava hitro preveri, ali je uporabnik v neposredni bližini dogodka.

Ob poslanem sporočilu od povezovalnih strežnikov dobimo rezultat pošiljanja, na podlagi katerega lahko izvemo, ali je bilo pošiljanje uspešno, in ali je morda kateri izmed uporabnikov našo aplikacijo odstranil.

Za pošiljanje sporočil mora naš strežnik imeti dovoljenje. To dovoljenje pridobimo tako, da v Google-konzoli za razvijalce, kjer smo registrirali našo aplikacijo in

omogočili obveščanje z Google Cloud Messaging, navedemo naš strežnik na seznamu dovoljenih naslovov IP, ki lahko pošiljajo obvestila. Tako pridobimo poseben ključ, s katerim se naš strežnik ob pošiljanju sporočil predstavi povezovalnim strežnikom. Temu ključu pravimo tudi avtorizacijski žeton pošiljatelja.

4.2 Mobilna aplikacija

Zaradi potrebe po hitrem in prijaznem dostopu do informacij, ki nas zanimajo, smo se odločili izdelati mobilno aplikacijo. Aplikacija skrbi za posredovanje informacij o izrednih dogodkih, ki zanimajo naše uporabnike oziroma naročnike. Prav tako pa s pomočjo obvestil poskuša simulirati pozivnik, in tako uporabnike razbremeniti še enega kosa opreme ter neprekinjenega poizvedovanja po novih dogodkih. Za dodano vrednost pa naša aplikacija poskuša pri obveščanju upoštevati še uporabnikov kontekst.

Mobilna aplikacija je tisti del sistema, ki je namenjen in viden končnemu uporabniku 4.3. Napisana je za pametne telefone z Android operacijskim sistemom. Aplikacija je spodaj predstavljena z diagramom uporabe 4.4, v nadaljevanju podpoglavja pa sledi še podrobnejši opis.

4.2.1 Namestitev aplikacije

Ob namestitvi naša aplikacija zahteva pravice za uporabo interneta, stanja internetne povezave, za pisanje v zunanji pomnilnik naprave, za dostop do lokacije, za dostop do Google-računa, s katerim je naprava registrirana, in za bujenje naprave ob prejemanju obvestil ter za uporabo vibre.

Pravice so definirane v Android Manifestu, za namestitev pa se uporabnik z njimi mora strinjati 4.5.

V Android Manifestu so prav tako nastavljene pravice, potrebne za delovanje obveščanja z Google Cloud Messaging in intent, ki služi kot sprejemnik za obvestila.

4.2.2 Naročanje na obvestila

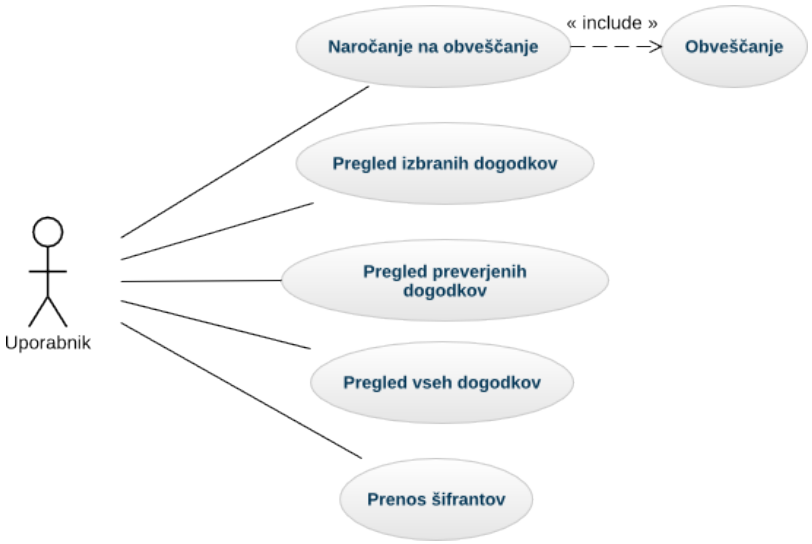
Ena izmed ključnih funkcionalnosti mobilne aplikacije je naročanje na obveščanje. Funkcionalnost je implementirana s pomočjo razreda *PreferenceFragment*, ki je

Slika 4.3: Osrednji meni aplikacije.

posebej zasnovan za gradnjo nastavitev aplikacije Android.

Ob prvem zagonu aplikacije se s strežniškega dela s pomočjo spletne storitve prenesejo šifranti, ti se zapišejo v zunanji pomnilnik naprave, ob odprtju nastavitev pa se iz pomnilnika preberejo in dinamično zgradijo v pregled z nastavitvami. Uporabnik lahko kadarkoli kasneje s pomočjo posebne akcije zahteva ponoven prenos šifrantov. Na ta način smo dosegli neodvisnost mobilne aplikacije od sprememb na šifrantih.

Vse nastavitve se avtomatsko shranjujejo v prostor za nastavitve aplikacije, s pritiskom na gumb "Naroči obveščanje" pa pošljemo nastavitve še na strežnik 4.6.



Slika 4.4: Diagram UML uporabe mobilne aplikacije.

4.2.3 Registracija naprave

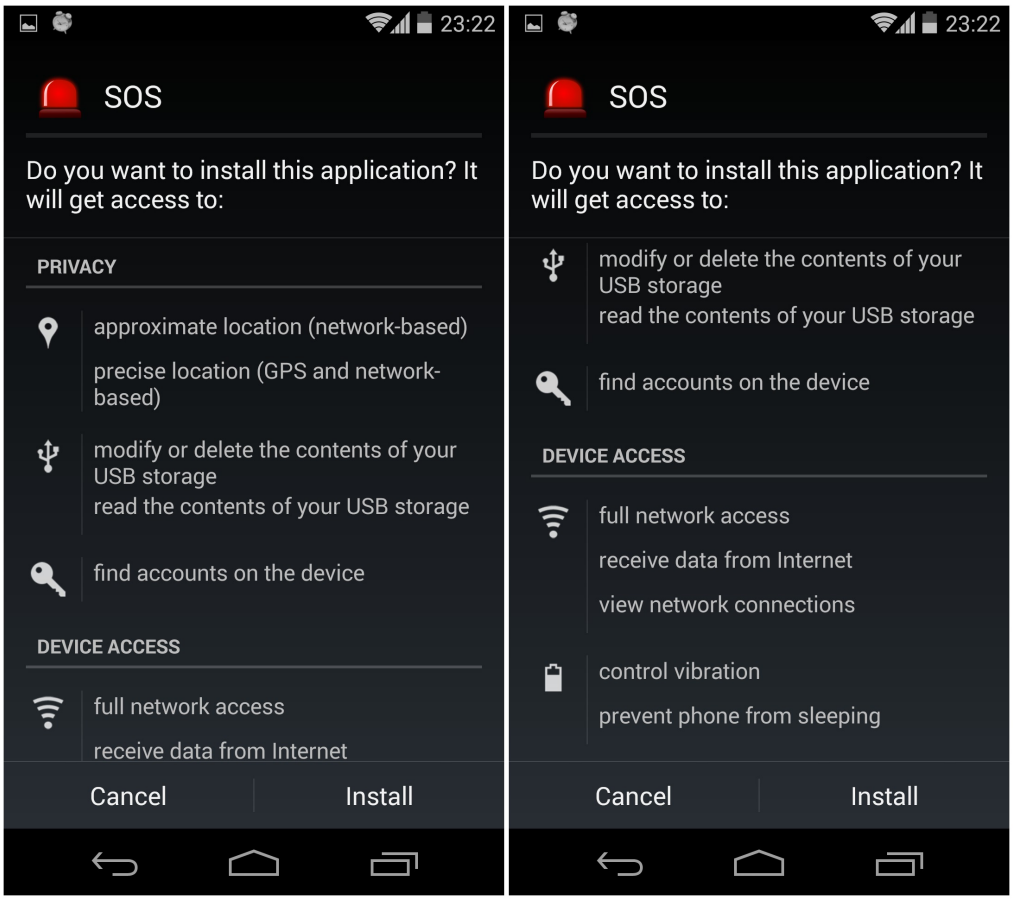
Ob pošiljanju nastavitev za obveščanje na strežnik poteka tudi registracija naprave za obveščanje z Google Cloud Messaging. Aplikacija pošlje svoj enoličen registracijski ključ strežniškem delu, pri tem pa si ga shrani v pomnilnik telefona in s tem zabeleži, da je bil pridobljen in poslan. Strežniški del ključ shrani in ga kasneje uporabi za naslavljanje naprave ob pošiljanju obvestil.

4.2.4 Informacija o lokaciji

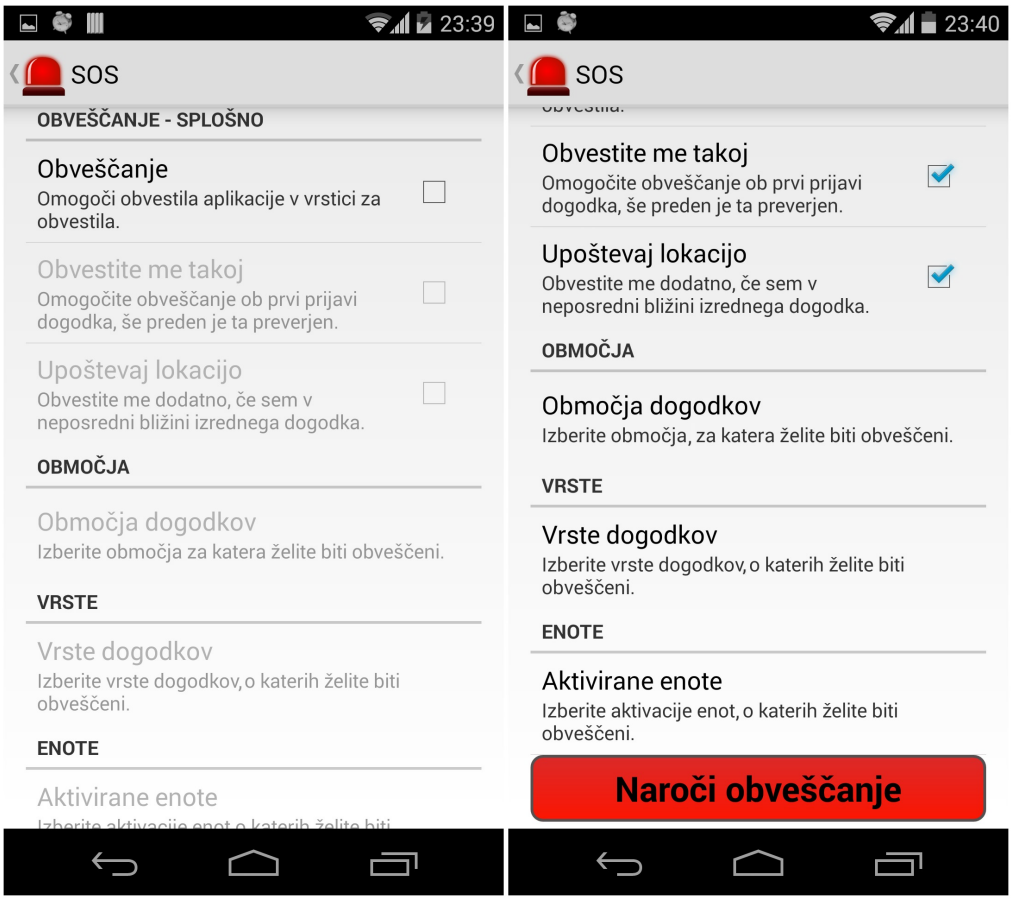
Kot smo že omenili, naš sistem upošteva uporabnikov fizični kontekst z informacijo o lokaciji. Če ima uporabnik omogočeno upoštevanje lokacije, se poleg obvestila o novih izrednih dogodkih pošljejo tudi imena lokacij dogodkov.

Intent, ki se proži ob prejemu obvestila, aktivira preverjanje lokacije. Dejansko lokacijo na podlagi njenega imena oziroma naslova ugotovimo s pomočjo metode *getFromLocationName* razreda *Geocoder* paketa *android.location*.

Dejansko bližino uporabnika pa preverimo s pomočjo nastavljanja opozorila bližine (angl. Proximity alert). Metodi *addProximityAlert* razreda *LocationManager* podamo lokacijo, radij, čas, po katerem naj poteče, in intent, ki se naj ob



Slika 4.5: Pogoji uporabe aplikacije.



Slika 4.6: Glavni pogled na nastavitve za naročanje na obveščanje.

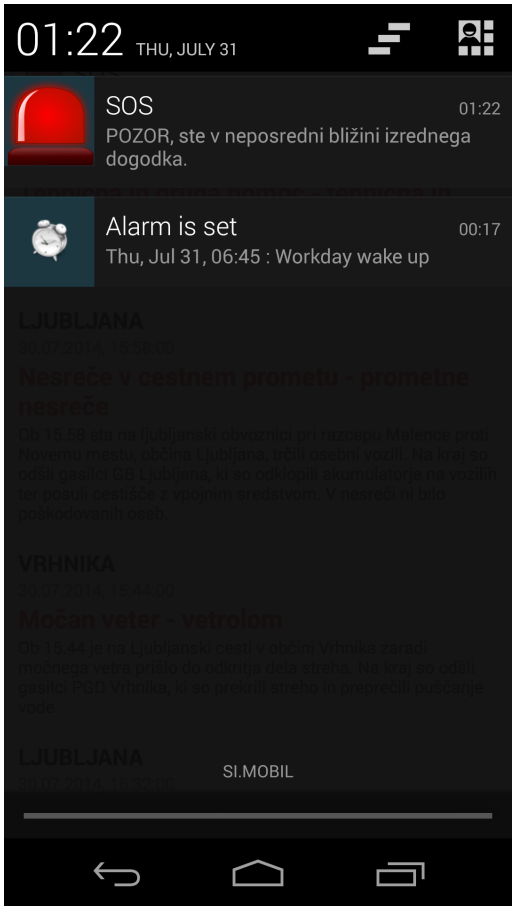
ugotovljeni bližini izvede. Ob ugotovljeni bližini se tako kliče intent, ki poskrbi za ustrezno opozorilo 4.7, ki obvesti uporabnika, da je v neposredni bližini izrednega dogodka, aplikacija pa samodejno prikaže najnovejše izbrane dogodke 4.8.

Izvorna koda 4.5: Obvestilo ob neposredni bližini izrednega dogodka.

```
public class ProximityActivity extends Activity {
    public static final int NOTIFICATION_ID = 1;
    private NotificationManager notificationManager;
    private static final long[] vibratePattern = {1000, 1000, 1000, 1000, 1000};

    @Override protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        notificationManager = (NotificationManager)this.getSystemService(Context.
             NOTIFICATION_SERVICE);

        PendingIntent contentIntent = PendingIntent.getActivity(
                this, 0, new Intent(this, EventsActivity.class), 0);

        NotificationCompat.Builder builder =
                new NotificationCompat.Builder(this).setSmallIcon(R.drawable.lilulilu)
                        .setContentTitle(getString(R.string.app_name))
                        .setStyle(new NotificationCompat.BigTextStyle().bigText("POZOR, ste v
                             neposredni blizini izrednega dogodka."))
                        .setVibrate(vibratePattern)
                        .setPriority(IntentFilter.SYSTEM_HIGH_PRIORITY)
                        .setLights(0xff0000, 1000, 1000);

        builder.setContentIntent(contentIntent);
        notificationManager.notify(NOTIFICATION_ID, builder.build());

        Intent intent = new Intent(this, EventsActivity.class);
        startActivity(intent);
    }
}
```

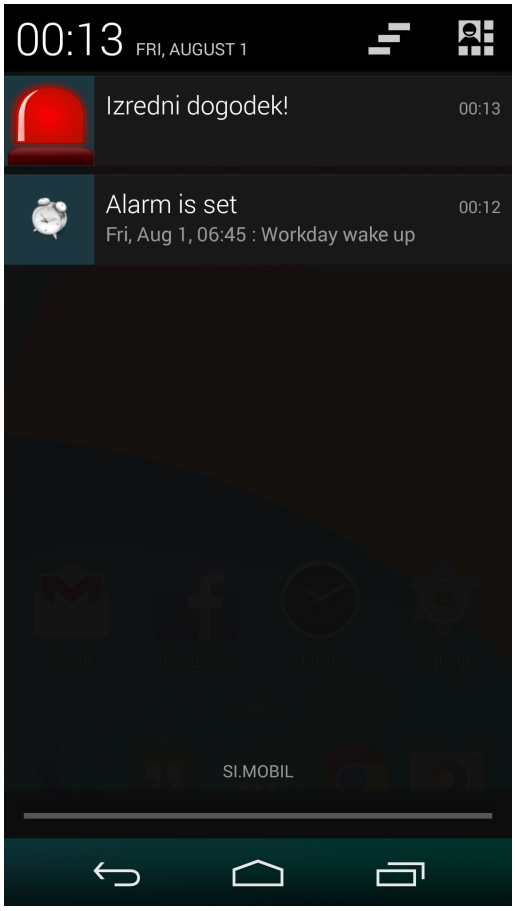
4.2.5 Obvestila

Kot smo že omenili, obvestila ne nosijo veliko podatkov, ampak služijo bolj kot opomnik za posodobitev podatkov o dogodkih 4.9. Ob dotiku obvestila se nam odpre, in s tem osveži pregled najnovejših dogodkov, na katere smo naročeni.



Slika 4.7: Obvestilo ob neposredni bližini izrednega dogodka.

Slika 4.8: Zaslonski sliki pregleda najnovejših nepreverjenih (levo) in preverjenih (desno) izrednih dogodkov.



Slika 4.9: Obvestilo ob izrednem dogodku.

Poglavje 5

Možne razširitve in izboljšave

Pri razvoju strežniškega dela in mobilne aplikacije smo se držali načel dobrega programiranja. Verjamemo, da so urejenost, modularnost, generičnost in dokumentiranost kode predpogoj za kakršne koli razširitve in izboljšave.

Že v času razvoja smo aplikacijo ponudili nekaj prostovoljcem, ki so veliko pripomogli k zgodnjem odkrivanju napak in njihovem hitrem odpravljanju. Aplikacijo smo testirali tako na zmogljivejših kot na povprečnih napravah, saj verjamemo, da slednje komaj pokažejo pravo kvaliteto aplikacije. S hitrostjo izvajanja aplikacije smo večinoma zadovoljni, bi pa ta seveda lahko bila še hitrejša. Uporabili bi lahko tudi nekatere prijeme, ki naredijo ugodnejšo izkušnjo pri tistih delih, ki jih težko pohitrimo in niso odvisni samo od naše implementacije. Pri nalaganju podatkov iz sistema SPIN bi lahko recimo uporabnika zamotili s prikazom zadnjih dogodkov, ki jih je prenesel, saj tako deluje aplikacija odzivnejša, kot če samo prikazuje indikator za delo. Pregledi dogodkov bi prav tako lahko uporabljali asinhrono nalaganje podatkov in tako hitro prikazali za en zaslon dogodkov, medtem ko bi se ostali naložili med drsenjem po pregledu.

Razmislili in dodali bi lahko tudi še kakšno poslovno pravilo in upoštevali še kakšen drugi vidik uporabnikovega konteksta.

Več izboljšav pa bi potreboval strežniški del. Preveriti bi bilo potrebno, kako se obnaša ob sočasnih zahtevah več uporabnikov, saj usluge spletnih storitev zaenkrat še niso implementirane asinhrono ali večnitno in so časovno precej zahtevne.

Pri obeh delih sistema je torej prostor za tehnične izboljšave, prav tako pa je prostor tudi za izboljšave z razširitvami funkcionalnosti.

Zaenkrat je funkcionalnosti bolj malo, so pa te pazljivo izbrane. Kot vemo, je potrebno aplikacijo na trg spraviti čim hitreje in dodatne funkcionalnosti implementirati v skladu z željami in potrebami uporabnikov. Predolg začetni razvoj velikokrat pripelje do tega, da nas kdo prehiti ali pa da implementiramo funkcionalnosti, ki se nam zdijo pomembne, medtem ko ima dejanski končni uporabnik čisto drugačne želje.

Poglavje 6

Zaključek

V sklopu diplomskega dela smo izdelali prototip sistema za obveščanje o izrednih dogodkih. Sistem je sestavljen iz strežniškega in mobilnega dela, podatke pa smo črpali s spletne aplikacije SPIN.

Strežniški del temelji na priznanih odprtokodnih tehnologijah in nudi storitve, potrebne za obveščanje uporabnikov mobilne aplikacije. Njegove ključne naloge so registracija uporabnikov, poizvedovanje po novostih s pomočjo spletne storitve iz sistema SPIN, in obveščanje naročenih uporabnikov v skladu z njihovimi željami. Zavedamo se, da bi za veliko število uporabnikov morali delovanje strežniškega dela izboljšati in zagotoviti večjo skalabilnost z asinhronostjo in večnitnostjo poizvedovanja, procesiranja novic in pošiljanja obvestil. Seveda pa je smiselno najprej preveriti, če je za našo aplikacijo sploh kakšno zanimanje.

Mobilna aplikacija ponuja uporabnikom priročen vpogled v izredne dogodke, objavljene na spletnem portalu SPIN, brez potrebe po obisku spletne aplikacije ali uporabe vira RSS in s tem dodatnega bralnika. Uporabnikom nudi tudi naročanje na obveščanje po meri v vrstici za obvestila, in jih tako razbremeni še poizvedovanja po novostih. Na željo uporabnikov aplikacija upošteva tudi njihovo lokacijo in jih v primeru neposredne bližine izrednega dogodka o tem tudi obvesti.

Še enkrat pa naj poudarimo, da je sistem v zgodnji fazi razvoja, in nikakor ni nadomestilo za pozivnik ter lahko služi le kot dopolnilo.

Literatura

[1] (2009) TIOBE Programming Community Index. Dostopno na: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

[2] (2014) Programming Language Popularity. Dostopno na: http://www.langpop.com/

[3] (2012) Differences between Java EE and Java SE - Your First Cup: An Introduction to the Java EE Platform. Dostopno na: http://docs.oracle.com/javaee/6/firstcup/doc/gkhoy.html

[4] (2014) Java Platform, Enterprise Edition. Dostopno na: http://en.wikipedia.org/wiki/Java_Platform,_Enterprise_Edition

[5] Bauer Christian, King Gavin, "Hibernate in Action", Manning, 2005, str. 5-23

[6] Debu Panda, Reza Rahman, Derek Lane, "EJB 3 in Action", Manning, 2007, str. 5-6

[7] Deitel Paul, Deitel Harvey, "Java How To Program, Eight Edition", Pearson Education, Inc., 2010, str. 1347

[8] (2014) HIBERNATE - Relational, Persistence for Idiomatic Java, Hibernate Reference Documentation, 3.3.1. Dostopno na: http://docs.jboss.org/jbossclustering/hibernate-jbosscache-guide-3.pdf

[9] dr. Rok Rupnik, dr. Cyprian Laskowski, Atrej Gognjavec, "Implementacija in preizkus dostave relevantnih podatkov" (November, 2013)

[10] (2014) Mobile app. Dostopno na: http://en.wikipedia.org/wiki/Mobile_app

[11] W. Frank Ableson, Robi Sen, Chris King and C. Enrique Ortiz, "Android in Action, Third Edition", Manning, 2011, str. 4

[12] (2014) Android, the world's most popular mobile platform. Dostopno na: http://developer.android.com/about/index.html

[13] (2014) Google Play Services. Dostopno na: https://developer.android.com/google/play-services/index.html

[14] (2014) Google Cloud Messaging, Key Concepts. Dostopno na: http://developer.android.com/google/gcm/gcm.html#key

[15] (2014) Google Cloud Messaging, Overview. Dostopno na: http://developer.android.com/google/gcm/gcm.html

[16] (2014) Google Cloud Messaging, Send a message. Dostopno na: http://developer.android.com/google/gcm/gcm.html#push-process

[17] (2014) Google Cloud Messaging, Recieve a message. Dostopno na: http://developer.android.com/google/gcm/gcm.html#receiving

[18] (2014) What is a GPS? How does it work? Dostopno na: http://www.loc.gov/rr/scitech/mysteries/global.html

[19] (2014) Location APIs. Dostopno na: http://developer.android.com/google/play-services/location.html

[20] (2014) LocationManager. Dostopno na: http://developer.android.com/reference/android/location/LocationManager.html

[21] (2014) RESTEasy. Dostopno na: http://resteasy.jboss.org/

[22] (2014) SPIN. Dostopno na: http://logos.si/

[23] (2014) SOAP Version 1.2. Dostopno na: http://www.w3.org/TR/2001/WDsoap1220010709/